UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/670,068	09/23/2003	Akihiko Tozawa	JP920020151JP1	5679

36380 7590 12/12/2006

RICHARD M. GOLDMAN
371 ELAN VILLAGE LANE
SUITE 208, CA 95134

EXAMINER	
VAUTROT, DENNIS L	

ART UNIT	PAPER NUMBER
2167	

DATE MAILED: 12/12/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/670,068	TOZAWA ET AL.	
	Examiner	Art Unit	
	Dennis L. Vautrot	2167	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *9/28/2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-6,10,11,14,15 and 18-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6,10,11,14,15 and 18-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *23 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Amendment

1. The applicants' amendment, filed 28 September 2006, has been received, entered into the record and considered.

2. As a result of the amendment; claims 1, 2, 4, 10, 14, 18, 20, and 22 are amended. Claims 7-9, 12-13, and 16-17 are canceled. Claims 1 – 6, 10 – 11, 14 – 15, and 18 – 22 are pending in the application.

3. The amended independent claims' rejections have been changed to include a further rejection based of the newly inserted language.

Specification

4. In light of the amendment to the specification adding the claim for foreign priority, the objection for the specification has been withdrawn.

Claim Rejections - 35 USC § 101

5. The rejection for Claims 1 - 22 under 35 U.S.C. 101 has been withdrawn.

Claim Rejections - 35 USC § 112

6. The rejection for claims 12 - 19 based on insufficient antecedent basis for this limitation in the claim is withdrawn.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

8. Claims 1, 4, 7, and 10 are rejected under 35 U.S.C. 102(b) as being anticipated by the article "Efficient Filtering of XML Documents for Selective Dissemination of Information", International Conference on Very Large Data Bases, 2000, by Altinel, Mehmet and Franklin, Michael J., (Hereinafter, **Altinel, et al.**)

9. Regarding claim 1, **Altinel, et al.** teaches a document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:

a compiling device for generating a query automaton by storing an input query expression, performing parsing, identifying different types of nodes in said element identifiers (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries..." and see pages 56-57 "Each XPath query is decomposed into a set of path nodes by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query." In other words, the documents are converted into the query automaton, identified, and stored.);

a query automaton storage device for storing the query automaton generated by said compiling device (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries..."); and

a query automaton evaluator for reading out said query automaton from said storage device and storing said automaton (See page 54, column 1 "These profiles are 'standing queries,' which are (conceptually) applied to all incoming documents" In other words "applied to all incoming documents" is equivalent to "reading out."), while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton and outputting the searched node (See page 54, column 1 "The other key inputs to an SDI system are the documents to be filtered."; also see page 57, column 2 "We use an XML parser that is based on the SAX interface, which is a standard for event-based XML parsing..." and see page 56, column 2 "The Query Index is built over the states of the XPath queries." The SAX interface from the reference uses the stream search as described in the specification on page 3, line 1.) and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are

interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.)

10. Regarding claim 4, **Altinel, et al.** teaches a document-searching method for searching a document having a hierarchical structure with elements separated by element identifiers, comprising the steps of:

generating a query automaton by storing a query expression input by a compiling device, performing parsing, and identifying different types of nodes in said element identifiers (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries..." and see page 56-57 "Each XPath query is decomposed into a set of path nodes by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query.");

storing the query automaton generated by said compiling device in a query automaton storage device (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the

queries..." In other words, the documents are converted into the query automaton, identified, and stored.); and

reading out said query automaton from said query automation storage device and storing said query automaton (See page 54, column 1 "These profiles are 'standing queries,' which are (conceptually) applied to all incoming documents"), while reading in said document and performing a stream search with a query automaton evaluator by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton. (See page 54, column 1 "The other key inputs to an SDI system are the documents to be filtered." also see page 57, column 2 "We use an XML parser that is based on the SAX interface, which is a standard for event-based XML parsing..." and see page 56, column 2 "The Query Index is built over the states of the XPath queries." The SAX interface from the reference uses the stream search as described in the specification on page 3, line 1.) and

storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node,

then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.)

11. Regarding claim 10, **Altinel, et al.** teaches a computer-readable storage medium storing a computer-executable program for performing a document-searching method for searching a document having a hierarchical structure with elements separated by element identifiers, wherein said program causes a computer to perform the steps of:

functioning as a compiling device for generating a query automaton by storing an input query expression, performing parsing, and identifying different types of nodes in said element identifiers (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries..." and see page 56-57 "Each XPath query is decomposed into a set of path nodes by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query." In other words, the documents are converted into the query automaton, identified, and stored.);

storing a query automaton generated by said compiling device in a query automaton storage device (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries...");

functioning as a query automaton evaluator for reading out said query automaton from said storage device and storing said query automaton (See page 54, column 1 "These profiles are 'standing queries,' which are (conceptually) applied to all incoming documents"), while reading in said document and performing a stream search by using states of a plurality of different types of nodes in said element identifiers included in said document and said query automaton (See page 54, column 1 "The other key inputs to an SDI system are the documents to be filtered." also see page 57, column 2 "We use an XML parser that is based on the SAX interface, which is a standard for event-based XML parsing..." and see page 56, column 2 "The Query Index is built over the states of the XPath queries." The SAX interface from the reference uses the stream search as described in the specification on page 3, line 1.) and

storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the

promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.)

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 2 and 3 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Altinel et al.** as applied to claim 1 above, and further in view of Yannis Papakonstantinou and Victor Vianu "DTD Interference from Views of XML Data", POD 2000, Dallas, TX. (Hereinafter, **Vianu, et al**.).

14. Regarding claim 2, **Altinel et al.** teaches a document searching system substantially as claimed. **Altinel et al.** fails to teach said query automaton evaluator determines a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node. However, **Vianu et al.** teaches said query automaton evaluator determines a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search

result of said left node and said lower node. (See page 35, column 2, last paragraph and page 36, column 1, first paragraph "Queries extract variable bindings from the input using a tree pattern involving regular expressions to navigate both vertically and horizontally. Horizontal regular expressions provide a powerful way to query the order of the nodes. They are encountered in some semistructured query languages and seamlessly couple with the more commonly used regular path expressions for vertical navigation.") It would have been obvious to one with ordinary skill in the art to combine the search method of **Altinel et al.** with the idea of storing information about the left node and lower node, as in **Vianu et al.** because of the time saving efficiency of having the information stored with the element identifier. It is for this reason that one of ordinary skill in the art would have been motivated to have said query automaton evaluator determine a state transition of a node under determination by storing a left node and a lower node in correspondence with an identified element identifier, and evaluate said query automaton with a search result of said left node and said lower node.

15. Regarding claim 3, **Altinel et al.** teaches a document searching system substantially as claimed. **Altinel et al.** fails to teach said compiling device generates a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. However, **Vianu et al**. teaches said compiling device generates a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. (See page 37, column 1, last

paragraph "...has a finite set of *Q* states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include the said compiling device which generates a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon.

16. Claims 5 and 6 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Altinel et al.** as applied to claim 4 above, and further in view of Yannis Papakonstantinou and Victor Vianu "DTD Interference from Views of XML Data", POD 2000, Dallas, TX. (Hereinafter, **Vianu, et al**.).

17. Regarding claim 5, **Altinel et al.** teaches a document searching system substantially as claimed. **Altinel et al.** fails to teach performing a stream search comprises a step of determining a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left

node and said lower node. However, **Vianu et al.** teaches performing a stream search comprises a step of determining a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node. (See page 35, column 2, last paragraph and page 36, column 1, first paragraph "Queries extract variable bindings from the input using a tree pattern involving regular expressions to navigate both vertically and horizontally. Horizontal regular expressions provide a powerful way to query the order of the nodes. They are encountered in some semistructured query languages and seamlessly couple with the more commonly used regular path expressions for vertical navigation.") It would have been obvious to one with ordinary skill in the art to combine the search method of **Altinel et al.** with the idea of storing information about the left node and lower node, as in **Vianu et al.** because it provides the advantage of evaluating relationships which cannot be evaluated in a conventional stream search because of the additional information. It is for this reason that one of ordinary skill in the art would have been motivated to perform a stream search that comprises a step of determining a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node.

18. Regarding claim 6, **Altinel et al.** teaches a document searching system substantially as claimed. **Altinel et al.** fails to teach said step of generating a query

automaton comprises a step of generating a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. However, **Vianu et al**. teaches said step of generating a query automaton comprises a step of generating a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. (See page 37, column 1, last paragraph "...has a finite set of Q states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to have said step of generating a query automaton comprises a step of generating a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon.

19. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Altinel et al.** as applied to claim 10 above, and further in view of Yannis Papakonstantinou and Victor Vianu "DTD Interference from Views of XML Data", POD 2000, Dallas, TX. (Hereinafter, **Vianu, et al.**). **Altinel et al.** teaches a document searching system

substantially as claimed. **Altinel et al.** fails to teach said performance of a stream search determines a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node, and wherein said query automaton is generated as a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. However, **Vianu, et al.** teaches said performance of a stream search determines a state transition of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier, and evaluating said query automaton with a search result of said left node and said lower node; (See page 35, column 2, last paragraph and page 36, column 1, first paragraph "Queries extract variable bindings from the input using a tree pattern involving regular expressions to navigate both vertically and horizontally. Horizontal regular expressions provide a powerful way to query the order of the nodes. They are encountered in some semistructured query languages and seamlessly couple with the more commonly used regular path expressions for vertical navigation.") It would have been obvious to one with ordinary skill in the art to combine the search method of **Altinel et al.** with the idea of storing information about the left node and lower node, as in **Vianu et al.** because it provides the advantage of evaluating relationships which cannot be evaluated in a conventional stream search because of the additional information. It is for this reason that one of ordinary skill in the art would have been motivated to include said performance of a stream search determines a state transition

of a node under determination at the moment by storing a left node and a lower node in correspondence with an identified element identifier;

and wherein said query automaton is generated as a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon. (See page 37, column 1, last paragraph "...has a finite set of *Q* states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include said query automaton is generated as a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon.

20. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Gupta et al.** (US Patent Application Publication 2004/0034651) in view of Yannis Papakonstantinou and Victor Vianu "DTD Interference from Views of XML Data", POD 2000, Dallas, TX. (Hereinafter, **Vianu, et al.**) and further in view of **Altinel et al.**

Gupta et al. teaches a compiling method for generating a query automaton for performing a document search, comprising the steps of:

generating and registering a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in terms of search, and storing a plurality of states of a backward node in correspondence with said backward node into a query automaton storage device; (See page 15, paragraph [0266] "Technically, [it] is straightforward to implement as it amounts to a simple syntactic transformation from XML elements to F—Logic expressions (e.g., using an XML parser whose output is "pretty-printed" to F-Logic, ... Here examiner interprets the first part of the claim to mean that each node of the hierarchically written document is interpreted into a logical expression as is done in the reference).

Gupta et al. does not teach generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device and storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

However, **Vianu, et al.** teaches generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device. (See page 37, column 1, last paragraph "...has a finite set of Q states including a

distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device.

Additionally, **Altinel et al.** teaches storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its

corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.) It would have been obvious to one with ordinary skill in the art at the time of the invention to combine the teachings of **Gupta** and **Vianu** with that of **Altinel** because all of the references are related to Querying variations of XML documents, and by including the separate storage means for storing the results of the query automaton evaluator, the method can be more efficient by having the results stored for later use. It is for this reason that one of ordinary skill in the art would have been motivated to include storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

21. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Gupta et al.** in view of **Vianu, et al.** and in view of **Altinel** as applied to claim 14 above. **Gupta et al.** teaches a compiling device substantially as claimed. **Gupta et al.** fails to teach said compiling method comprises a step of identifying said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node. However, **Vianu, et al.** teaches said compiling method comprises a step of identifying said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node. (See page 39, column 2 "Intuitively, an edge label describes vertical navigation in the input

loto. For each node reached by vertical navigation, the node label describes horizontal navigation in the list of its children.") Examiner interprets this claim to mean that the states of the node are made up of the states of the left and lower nodes. It would have been obvious to one with ordinary skill in the art to store the states of the connected nodes in order to more efficiently search on hierarchical documents. It is for this reason that one of ordinary skill in the art would have been motivated to include said compiling method comprises a step of identifying said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node.

22. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Gupta et al.** (US Patent Application Publication 2004/0034651) in view of Yannis Papakonstantinou and Victor Vianu "DTD Interference from Views of XML Data", POD 2000, Dallas, TX. (Hereinafter, **Vianu, et al.**) and in view of **Altinel**.

Gupta et al. teaches a computer-readable storage medium storing a program for causing a computer to perform a compiling method for generating a query automaton for performing a document search, wherein said program causes a computer to perform the steps of:

generating and registering a state transition by replacing an axis including an axis in the opposite direction and a logical expression including a conjunction or a negative expression while keeping an input query expression equal in term of search, and storing the plurality of states of said backward node in correspondence with said

backward node into a storage device (See page 15, paragraph [0266] "Technically, [it] is straightforward to implement as it amounts to a simple syntactic transformation from XML elements to F—Logic expressions (e.g., using an XML parser whose output is "pretty-printed" to F-Logic, ... Here examiner interprets the first part of the claim to mean that each node of the hierarchically written document is interpreted into a logical expression as is done in the reference).

Gupta et al. does not teach generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device and storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

However, **Vianu, et al.** teaches generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device. (See page 37, column 1, last paragraph "...has a finite set of *Q* states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned; with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in

order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include generating a query automaton by registering a plurality of states of said backward node, a condition for transition, at least a search state, and a reached state in correspondence with each other in said storage device.

Additionally, **Altinel et al.** teaches storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.) It would have been obvious to one with ordinary skill in the art at the time of the invention to combine the teachings of **Gupta** and **Vianu** with that of **Altinel** because all of the references are related to Querying variations of XML documents, and by including the separate storage means for storing the results of the query automaton evaluator, the

method can be more efficient by having the results stored for later use. It is for this reason that one of ordinary skill in the art would have been motivated to include storing the output of the query automaton evaluator in a search result storage means for, and thereafter outputting the stored output of the query automaton evaluator and the output of the searched node.

23. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Gupta et al.** in view of **Vianu, et al.** and in view of **Altinel** as applied to claim 18 above. **Gupta et al.** teaches a compiling device substantially as claimed. **Gupta et al.** fails to teach said program comprises a step of causing a computer to identify said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node. However, **Vianu, et al.** said program comprises a step of causing a computer to identify said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node. (See page 39, column 2 "Intuitively, an edge label describes vertical navigation in the input loto. For each node reached by vertical navigation, the node label describes horizontal navigation in the list of its children.") Examiner interprets this claim to mean that the states of the node are made up of the states of the left and lower nodes. It would have been obvious to one with ordinary skill in the art to store the states of the connected nodes in order to more efficiently search on hierarchical documents. It is for this reason that one of ordinary skill in the art would have been motivated to include

said program comprises a step of causing a computer to identify said backward node as a left node or a lower node according to a type of said element identifier, and wherein said plurality of states are states of said left node and said lower node.

24. Claims 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Altinel, et al.** in view of **Vianu et al.**

25. Regarding claim 20, **Altinel, et al.** teaches a document-searching system for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:

a compiling device for generating a two-state input automaton for enabling a state transition by storing an input query expression, performing parsing, (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries..." and see page 56-57 "Each XPath query is decomposed into a set of path nodes by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query." In other words, the documents are converted into the query automaton and stored.);

an automaton query storage device for storing said two-state input automaton (See page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries...");

an automaton-evaluating device for enabling a state transition by reading out two-state input automaton from said storage device and storing said automaton (See page 54, column 1 "These profiles are 'standing queries,' which are (conceptually) applied to all incoming documents"); while reading in said document and identifying said two states (See page 54, column 1 "The other key inputs to an SDI system are the documents to be filtered."; also see page 57, column 2 "We use an XML parser that is based on the SAX interface, which is a standard for event-based XML parsing..." and see page 56, column 2 "The Query Index is built over the states of the XPath queries." The SAX interface from the reference uses the stream search as described in the specification on page 3, line 1.) and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note

that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.)

Altinel, et al. fails to teach reading at least two states assigned to different types of nodes in said element identifiers.

However, **Vianu, et al.** teaches reading at least two states assigned to different types of nodes in said element identifiers. (See page 35, second column last paragraph "Queries extract variable bindings from the input using a tree pattern involving regular expressions to navigate both vertically and horizontally." Examiner interprets the two states to mean the left node and the lower node as disclosed in the specification.) One with ordinary skill in the art would have recognized the advantage of combining the document searching system as disclosed in **Altinel, et al.** with the two-states as disclosed in **Vianu, et al.** for its speed and time efficiency. It is for this reason that one of ordinary skill in the art would have been motivated to include reading at least two states assigned to different types of nodes in said element identifiers.

26. Regarding claim 21, **Altinel, et al.** teaches a document searching system substantially as claimed. **Altinel, et al**. fails to teach said two states are states of a left node and a lower node of a tree structure generated in correspondence with an identified element identifier, and wherein said two-state input automaton uses three states of said automaton-evaluating device. However, **Vianu, et al.** teaches said two states are states of a left node and a lower node of a tree structure generated in correspondence with an identified element identifier, and wherein said two-state input

automaton uses three states of said automaton-evaluating device. (See page 35, second column, last paragraph "Queries extract variable bindings from the input using a tree pattern involving regular expressions to navigate both vertically and horizontally." Examiner interprets the two states to mean the left node and the lower node as disclosed in the specification.; and See page 37, column 1, last paragraph "...has a finite set of Q states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of having the two states of the left and lower node in order to individually evaluate each node more efficiently as well as including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include said query automaton is generated as a query automaton with a state transition corresponding to an initial state, a final state, and a search state registered thereon.

27. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Altinel, et al.** in view of **Vianu, et al. Altinel, et al.** teaches a query automaton evaluator for evaluating a query automaton for searching a document having a hierarchical structure with elements separated by element identifiers, comprising:

means for reading out a query automaton from a query automaton storage device that enables a plurality of inputs generated by a compiling device to be determined at a time and storing the query automaton (See page 54, column 1 "These profiles are 'standing queries,' which are (conceptually) applied to all incoming documents" In other words "applied to all incoming documents" is equivalent to "reading out." and see page 55, column 2 "In contrast, in SDI systems, large numbers of queries are stored, and the documents are individually mated to the queries...");

means for identifying a plurality of different types of inputs of said element identifiers included in said document (See pages 56-57 "Each XPath query is decomposed into a set of path nodes by the XPath parser. These path nodes represent the element nodes in the query and serve as the states of the FSM for the query.") and

a search result storage means for storing the output of the query automaton evaluator, and for thereafter outputting the stored output of the query automaton evaluator and the output of the searched node. (See page 57, column 2, 4th paragraph "All of the path nodes representing future states are stored in the Wait Lists of their respective element names. A state transition in the FSM of a query is represented by promoting a path node from the Wait List to the Candidate List." The lists here are interpreted to be the storage means for storing the output of the query automaton. And see page 58, first column, 3rd full paragraph "If this is the final path node of the query (i.e., its final state) then the document is deemed to match the query. Otherwise, if it is not the final node, then the query is moved into its next state. This is done by copying the next node for the query from its Wait List to its corresponding Candidate List (note

that a copy of the promoted node remains in the Wait List)." This determination represents the output of the query automaton evaluator.)

Altinel, et al. fails to teach means for assigning a state transition among three states including a search state by using said identified input and a plurality of inputs registered in said query automaton.

However, **Vianu et al.** teaches means for assigning a state transition among three states including a search state by using said identified input and a plurality of inputs registered in said query automaton (See page 37, column 1, last paragraph "...has a finite set of Q states including a distinguished initial state *q0* and an accepting state *qf*. In computation, the automaton labels the nodes of the tree with states, according to a set of rules, called transitions." Here the initial state is directly mentioned, with the accepting state being equivalent to the final state of the application and one of the other finite states not mentioned impliedly referring to the search state.) One with ordinary skill in the art would have recognized the advantage of including a state transition with the states mentioned in order to know which nodes to store (search), and which ones should be sent to an output process (final). It is for this reason that one of ordinary skill in the art would have been motivated to include means for assigning a state transition among three states including a search state by using said identified input and a plurality of inputs registered in said query automaton.

Conclusion

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dennis L. Vautrot whose telephone number is 571-272-2184. The examiner can normally be reached on Monday-Friday 9:00-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Cottingham can be reached on 571-272-7079. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Dv
4 December 2006

JOHN COTTINGHAM
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

L. S. Le